IIc Plus Preliminary Reference

November 1, 1988

Contents

Preface

4 About This Reference
4 About The IIc Plus

Part 1. Hardware and New Features

6 Compatibility With Previous Apple IIc Models
6 Overall Compatibility
6 Hardware Differences
8 Firmware Differences
8 Feature Differences
12 New Hardware
12 Internal Disk Drive
12 Internal Power Supply
12 Connectors
12 Keyboard
13 Accelerator Circuitry
13 Minor Circuitry Changes
14 Appendix A. Connector Pinout and Keyboard Layout
15 Figure A-1. IIc Plus Serial Mini-DIN Connectors (J6 and J12)
16 Figure A-2. IIc Plus External Disk Drive Connector (J2)
17 Figure A-3. IIc Plus Internal Drive Connector (J8)
18 Figure A-4. IIc Plus Power Supply Connector (Internal – J1)
19 Figure A-5. IIc Plus Keyboard Connector (Internal – J9)
20 Figure A-6. IIc Plus Disk Eject Switch Connector (Internal – J7)
21 Figure A-7. IIc Plus Keyboard Layout
22 Figure A-8. IIc Plus Expansion Connector (Internal – J14)
23 Figure A-9. IIc Plus Internal Modem Connector (J15)

Part 2. The Accelerator Feature

25 Introduction
25 Fast Versus Synchronous Speed
25 Synchronous Mode
25 Fast Mode
26 What Will and What Won't Work
26 Enabling the Accelerator
26 Disabling the Accelerator
27 Programming Hints
27 The ROM WAIT Routine
27 Booting Into Escape Mode
27 Temporarily Disabling the Accelerator Chip

Part 3. IIc Plus Firmware and 3.5" Drive Support

29 Overview
29 Goals
29 Hardware Requirements
29 Assumptions

30 Configurations
31 Startup (Boot) Sequence
31 ProDOS and Smartport Calls
31 Standard ProDOS Support
31 Smartport Support
33 Special Considerations
33 Disk II and Non-ProDOS Application
33 Differences Between the IIc Plus and previous Apple IIc's
34 IWM Reset on the IIc Plus
34 Limitations
34 Smartport Extensions
34 Copy Protection on Disk 3.5
35 Direct Access to Disk IIs
35 (Future) Higher Data Rate Drives

Addendum

37 Dealing with Reset
38 Blank
39 IIc Plus Smartport device identification
40 Mouse/VBL Handling & Accelerated Serial I/O
44 HAndling of Time-dependent Routines in Accelerated Mode
45 Identifying the IIc Plus

IMPORTANT

The IIc Plus is shipped with a "shipper" (dummy) disk installed in its drive(s). When power is initially turned on, the IIc Plus will hang for about 15 seconds before it ejects the "shipper" disk(s) and then will continue the startup process. This is normal operation and does not indicate any malfunction.

Preface

About These Preliminary Notes

This document defines the changes made to the Apple IIc for the IIc Plus project. This document is divided into three parts. Part 1 discusses compatibility issues, describes the new features of the IIc Plus, and provides pin-out diagrams for all new connectors. Part 2 describes the IIc Plus's new accelerator feature, provides programming hints to developers, and gives instructions for enabling and disabling the accelerator chip. Part 3 describes areas of the IIc Plus's firmware that are different (or operate differently) than previous versions of the Apple IIc.

These Preliminary Notes are provided as a service to third party developers and may contain information or specifications which are still under consideration by Apple. Apple makes no guarantee concerning the accuracy of these Preliminary Notes with respect to the final release of the product, nor should the recipients of this document construe such a guarantee. Apple reserves the right to change or delete any specification, product, service, or text contained or defined in these Preliminary Notes without prior notice. Development of products, services, or documentation on the basis of data contained in these Preliminary Notes is undertaken at the sole risk and expense of the developing party.

About the IIc Plus

The IIc Plus consists of an Apple IIc CPU with memory expansion capability, but with the following modification and new features:

* The 5.25" internal disk drive has been replaced with a 3.5" drive.
* A new 28 pin gate array and 2Kx8 static RAM have been added to the logic board as "glue logic" to internal and external Apple 3.5 Drives.
* A new 84 pin gate array accelerator chip, a 16 MHz Crystal Oscillator, and two 8Kx8 static RAMs have been added. This circuitry allows the computer to run much faster than the previous Apple IIc and permits user selectable fast/slow control of the speed.
* The keyboard has been replaced with the new Apple Standard Keyboard, without numeric keypad.
* The floor power supply has been replaced by an internal power supply.
* The rear handle will lock into place better than earlier versions of the IIC.
* The serial ports have been refitted as 8-pin mini-DINs.
* The headphone jack has been removed.
* The volume control has been relocated to above the keyboard as a slide potentiometer in place of the keyboard (Dvorak) switch.
* The keyboard (Dvorak) switch is replacing the 40/80 column switch; 40/80 column switching is now software-controlled, but the hardware default is 80 column.
* The standard Security Kit is supported.
* The case and keyboard are platinum colored.

Part 1

Hardware and New Features

Compatibility With Previous Apple IIc Models

Overall Compatibility

From a functional standpoint, there are very few areas which present any compatibility conflicts between previous versions of the Apple IIc and the IIc Plus. Virtually all software written for previous models will run on the IIc Plus, and none of the hardware changes should have any adverse impact on software, although the change in the internal disk drive (to 3.5") might encourage a change in the media on which such software is resident. External Apple 5.25 drives are supported as before, as well as Apple 3.5 (Unified) and UniDisk 3.5 Drives.

Applications that use the new accelerator feature will obtain a performance improvement that will vary depending upon the application. However, applications and hardware products that use DMA or some other critical timing code must run in the synchronous mode (1.024 MHz) and can not take advantage of the fast mode (4 MHz) capability of the computer. If the accelerator is enabled (fast mode) and an application or product does not appear to boot or function properly, you must disable the accelerator (synchronous mode) before running the application or product. If the application or product functions normally with the accelerator enabled, then it can be allowed to run in fast mode. An exception here might be an application with messages to the user where the message does not remain on the screen long enough for the user to finish reading it. In general, most applications and hardware products should run in fast mode without any problems. Refer to Part 2 of this preliminary note for a detailed description of the accelerator feature, programming hints, and instructions for enabling and disabling the accelerator.

Upgrade Compatibility: The IIc Plus supports memory expansion just like the current Apple IIc. Third party supplied memory expansion card designs for the previous versions of the Apple IIc can work in the IIc Plus if the cards are re-routed to interface to the available internal connectors. Note that many boards that worked by connecting to the MMU and CPU sockets will not work even though they still fit in the sockets. Specifically, any card or hardware device that used to plug into the 65C02 socket is likely to have problems since the 65C02 is running at 4 MHz and is not directly connected to the "Apple bus". However, since the IIc Plus logic board cannot be mounted in previous Apple IIc cases, and since the disk drive, power supply, and keyboard are different for the IIc Plus, previous Apple IIc systems cannot be upgraded to a IIc Plus for less than the cost of a new IIc Plus.

Hardware Differences

From a hardware compatibility standpoint, there are a few major differences between the IIc Plus and the Apple IIc with memory expansion, as follows:

* Disk drives/media
* Keyboard
* Rear Handle
* Connectors
* Power Supply

Disk Drives and Media

The IIc Plus supports both 3.5" and 5.25" disk drives. However, a 5.25" disk drive ***is not*** supplied with the IIc Plus; it is an optional external drive. This fact does present a compatability problem with previous versions of the IIc, inasmuch as software residing on 5.25" disks cannot be loaded into a IIc Plus machine without an external 5.25" drive connected. Software that is specifically written for the 5.25" internal drive Apple IIc will give the user confusing messages if it refers to the 5.25" drive as the "Built-in Drive" or "Internal Drive" on the IIc Plus, since the IIc Plus's internal drive is a 3.5" drive. Similarly, the same software might also refer to the internal drive as "External Drive 1" because the software thinks it is running on the previous Apple IIc. Although this software makes it confusing for the user, it will still run properly if the user can think in terms of the previous Apple IIc. Hopefully, these packages will be updated to remove nomenclature such as "Internal" or "External" or "Built-in."

Finally, since the internal 3.5" drive is similar to the Apple 3.5 Drive, some programs utilizing copy protection schemes specific to the UniDisk 3.5 Drive could have problems running on the IIc Plus.

Keyboard

The IIc Plus keyboard will use the Apple Universal Keyboard layout, and includes a slide potentiometer for speaker volume control in the position previously occupied by the "keyboard" switch (See Figure A-7). The "keyboard" switch now occupies the position previously occupied by the "40/80" column switch, which is now strictly under software control. Most programs heretofore did not use the "40/80" column switch.

Rear Handle

The IIc Plus handle will lock into place better than earlier versions of the Apple IIc. In this way, it can raise the rear portion of the unit when in use. However, the unit is not designed to be carried with the handle in the locked, right-angle position. Doing so would not only be an awkward way to carry the system, but it might also break the case at the hinge point of the handle. Carrying the system by the handle with the handle in the vertical position presents no problem.

External Connectors

The connectors used on the IIc Plus are not all compatible with those used on previous versions of the Apple IIc. Specifically, the two serial ports are now mini-DIN type connectors (Figure A-1). The serial ports are compatible with Apple IIGS serial interface cables and are compatible with RS232 and RS422 peripherals over short cable distances. However, the serial port signals still represent RS232.

The external disk drive connector remains compatible with disk drives connected to the previous Apple IIc, even though some signals have changed (Figure A-2). Compared to the previous Apple IIc, an EN2 signal has been addedand the Extint signal has been removed. In short, the DB19 connector signal pinout is now identical to that of the Apple IIGS. Elsewhere, the headphone jack has been removed, and the external power supply connector mates with a standard 3 pin power cord. The floor mount power supply is no longer required and is incompatible. Since the IIc Plus will be supplied with AC power

through this connector, battery packs for previous Apple IIc will be incompatible with the IIc Plus. A battery pack for the IIc Plus would have to supply the various voltages indicated on the internal power supply connector (Figure A-4), and would have to replace the internal power supply. The DC-DC converter of previous Apple IIc is not compatible with the internal IIc Plus power supply connector. The removal of the headphone jack and changes to the volume pot and internal power supply connector were necessary for manufacturability and serviceability of the IIc Plus.

Internal Connectors and Memory Expansion

The IIc Plus continues to support memory expansion (like the previous Apple IIc) through its 34 pin connector, J13. However, some third party cards may also need other signals such as those available from a new seven pin connector, J14 (located next to J13) and possibly from the MMU socket as well. Connecting to J14 and the MMU socket should not cause a problem. Popular third party memory card designs can continue to work if the card is re-routed to use J13. Connecting to the 65C02 socket is **not** recommended since the 65C02 is isolated from the Apple bus and runs at 4 MHz.

There are three additional internal connectors on the IIc Plus as compared to the previous Apple IIc. A seven pin connector, J15, located close to the power supply provides TTL level "ACIA" signals for possible internal modem designs. Such designs could use the security port for a bracket mounted telephone jack. The internal power supply connects to the logic board through a 7 pin straight header, J1, instead of the right-angle card-edge connector used to connect to the DC-DC converter of the previous Apple IIc. There is also a new internal two-pin connector, J7, on the logic board that connects to the disk drive eject button. The internal disk drive connector, the connector to the keyboard, and the memory card connector are still the same connectors, although some signals have changed on these connectors to meet the needs of the IIc Plus (See Figures A3 through A6).

Firmware Differences

Although the IIc Plus firmware retains all of the listed, supported entry points found in previous versions of the Apple IIc, the ROM map has been altered and new code has been added to handle disk I/O for Apple 3.5 Drives. Smartport calls are recommended for handling 3.5" disk I/O. Preliminary compatibility testing has shown that most existing Apple IIc software (whether booting off of the IIc Plus internal 3.5" disk drive or an external 5.25" disk drive) experience no problems on the IIc Plus. Further discussion of the IIc Plus firmware is provided in Part 3 of these Preliminary Notes.

Feature Differences

From a features standpoint, there are a number of differences between the IIc Plus and the prior version of the Apple IIc with memory expansion capability, as follows:

* Accelerator feature
* More External Drive Options
* Software Compatibility
* Boot Sequence
* 40/80 Column Switch
* Volume Control
* Security Kit Supported

Accelerator Feature

The IIc Plus includes a custom gate array accelerator circuit that controls the speed of the processor and allows the machine to run in either synchronous mode (1.024 MHz) or fast mode (4 MHz). The accelerator has been preset for the IIc Plus. On power up or cold boot, slots 1,2,5, and 6 are set to synchronous mode; slots 3, 4 and 7 are set to fast mode, and the speaker and the paddles are set to slow mode. The user cannot change this configuration. The accelerator feature permits the user to enable (fast mode) or disable (synchronous mode) the accelerator and control the operating speed of the computer. See Part 2 of these preliminary notes for more information on the accelerator feature.

More External Drive Options

The IIc Plus interfaces to the same external drives that the previous Apple IIc does and can also interface to the "unified" Apple 3.5 Drive. No more than three external drives, whether 3.5" drives or a mixture of 3.5" and 5.25" drives, are recommended due to power supply limits. Of these external disk drives, a maximum of two can be Apple 3.5 Drives or Apple 5.25 Drives.

The following table summarizes the maximum recommended disk drive configurations, their virtual slot/drive (S,D) assignments in software, and whether the drives are allowed to be booted from (B) or not (NB). In the table, A3.5=Apple 3.5 Drive, U3.5=UniDisk 3.5 Drive, Int.=Internal and Ext.=External:

IIc Plus	1st Drive in Chain	2nd Drive in Chain	3rd Drive in Chain
Int. 3.5 (S5,D1)	Ext. 5.25 (S6,D1,B)	Ext. 5.25 (S6,D2, NB)	
Int. 3.5 (S5,D1)	Ext. A3.5 (S5,D2,B)	Ext. A3.5 (S2,D1, NB)	Ext. 5.25 (S6,D1,B)
Int. 3.5 (S5,D1)	Ext. A3.5 (S5,D2,B)	Ext. A3.5 (S2,D1, NB)	Ext. U3.5 (S2,D2,NB)
Int. 3.5 (S5,D1)	Ext. A3.5 (S5,D2,B)	Ext. U3.5 (S2,D1, NB)	Ext. U3.5 (S2,D2,NB)
Int. 3.5 (S5,D1)	Ext. U3.5 (S5,D2,B)	Ext. U3.5 (S2,D1, NB)	Ext. 5.25 (S6,D1,B)
Int. 3.5 (S5,D1)	Ext. A3.5 (S5,D2,B)	Ext. 5.25 (S6,D1, B)	Ext. 5.25 (S6,D2,NB)
Int. 3.5 (S5,D1)	Ext. U3.5 (S5,D2,B)	Ext. 5.25 (S6,D1, B)	Ext. 5.25 (S6,D2,NB)

Although not explicit from the table, the internal 3.5 IIc Plus drive can be booted in all configurations. The IIc Plus supports both the UniDisk 3.5 and the Unified Apple 3.5 external disk drives in virtual slots 5 or 2 in accordance with ProDOS 8 and as defined in Apple's Smartport specification. The 5.25" external drive or drives are located in virtual slot 6.

It is also clear from the table that the 1st device in the chain can be booted from whether it is an Apple 3.5 Drive, a UniDisk 3.5 Drive, or a Apple 5.25 Drive. This is because there are two ***external*** slot/drive assignments that can be booted from: Slot 5, Drive 2 and Slot 6, Drive 1. In other words, the first external 3.5" drive and the first 5.25" drive can be booted from, in addition to the internal 3.5" drive. Compared to the previous Apple IIc, there is one new slot/drive assignment for booting from, namely, Slot 5, Drive 2.

When ***both*** 3.5" and 5.25" drives are being utilized, the 5.25" drives ***must*** be connected ***after*** the 3.5" drives, regardless of the type of 3.5" drive selected. It is ***not*** necessary to connect a 3.5" drive in order to connect a 5.25" drive. Also, if Apple 3.5 Drives are used, they have to be placed ***first*** in the chain, ahead of any UniDisk 3.5 Drives, and of course ahead of any 5.25" drives. This daisy chain order is very important if more than one drive is connected to the IIc Plus.

The internal drive for the IIc Plus is similar to the Unified style Apple 3.5 Drive in hardware interface, although not identical. Strictly speaking, the internal drive is a "bare" drive lacking the daisy chaining "mini" board and circuitry that is present in the Apple 3.5 Drive. On the logic board of the IIc Plus is a custom gate array and static RAM that allow the IIc Plus to us a Unified style 3.5" drive internally and also allow daisy chaining of Apple 3.5, UniDisk 3.5 and Apple 5.25 drives externally.

Software Compatibility

Software on 3.5" disks written for the previous Apple IIc could have problems on the IIc Plus if that software was written with too many assumptions about the Apple IIc. For instance, if copy protected software on 3.5" disk assumed that it would always be run on the "intelligent" UniDisk 3.5 Drive, this software **will** have problems when run on the IIc Plus's internal "unified" style drive. Similarly, if the software always expected it would be booted from a UniDisk 3.5 Drive in "slot 5, ***Drive 1***," then again this software is likely to have problems in booting from even an external UniDisk 3.5 Drive ("slot 5, ***Drive 2***") connected directly behind the IIc Plus.

If software follows Apple's guidelines of using the protocol converter (Smartport) in the Apple IIc firmware for talking to disk drives, if the software does not have to be run in a specific slot/drive assignment, and if the software does not make specific requirements on "screen hole" memory locations, that software is unlikely to have problems running on the IIc Plus. The only difference between the standard protocol converter (Smartport) specification and the IIc Plus is that the IIc Plus does not support any device specific control calls (extended calls). Also, most DOS 3.3 software is unlikely to have problems running on the IIc Plus since this portion of the firmware interface is minimally changed.

Also, as mentioned earlier in this section, software that uses nomenclature such as "Internal" or "Built-in" or "External" particular to the previous Apple IIc on the application's menu selections are likely to be confusing to the user.

Boot Sequence

The IIc Plus will boot from the memory expansion card ("Slot 4, Drive 1," if present), the internal 3.5" drive, the first external 3.5" drive, or the external 5.25" drive (if any), in that order. Note that the IIc Plus, like the current IIc, cannot boot from the memory expansion card in the event of a cold start. However, if properly formatted, the IIc Plus can boot from the card on a warm start.

Internal Hardware Compatibility

As mentioned earlier, many popular third party memory expansion cards will "fit" in the IIc Plus, but will not work due to the faster processor timing. It is recommended that these cards be adapted to connect to the internal connectors, and if necessary, the MMU socket as described previously under *Internal Connectors and Memory Expansion*. Some cards will try to map a ROM or some RAM into the Apple's memory map. This scheme will only work if it is done carefully. Developer's interested in doing this should contact Apple Technical Support.

40/80 Column Switch

The 40/80 Column Switch has been redefined to be the Keyboard Switch and labeled accordingly on the housing. 40/80 Column Switching is now executed in software only, by means of the soft switches. The default state is 80-column.

Volume Control

The Keyboard Switch has been redefined to be a slide volume control. As a result, the volume knob on the lower left side of the case has been removed.

Apple Security Kit

The IIc Plus supports the standard Apple Security Kit as implemented on the Apple IIGS.

New Hardware

Internal Disk Drive

The IIc Plus utilizes a 3.5" double-sided, double-density (800Kb) disk drive connected to the motherboard via a 20-pin ribbon cable. The interface circuitry to this drive and to other external 3.5" drives includes two new IC's on the Apple IIc logic board. The two IC's are as follows:

* 28 pin gate array "glue" chip
* 24 pin 2Kx8 static RAM chip

These two new IC's allow buffering of the data to and from the internal and external 3.5" drives, and provide the "glue logic" to interface to them. More details are provided Part 3 of these preliminary notes.

Internal Power Supply

The power supply utilized by the IIc Plus is a compact internal switching supply that provides up to 1.5 A at +5V, up to 0.9 A at +12V, and 0.1 A at –12V. This UL compliant power supply is rated at 18W total. Primary input voltage is directly connected to the transformer in order to isolate the motherboard and other unprotected contact points from high voltages.

Connectors

The IIc Plus utilizes two mini-DIN serial port connectors on the back panel and eliminates the headphone jack. Figure A-1 shows the pin-out diagram for this new serial connector. Also, the signals to the external drive DB-19 connector have changed slightly and Figure A-2 shows that pin-out diagram. Internally, the signals to the internal drive have changed somewhat (Figure A-3), and the connector for the power supply (to the logic board) is different from the card edge style of the DC-DC converter in the previous Apple IIc. The new power supply is connected by a 7 pin single row vertical header; the pin-out diagram is shown in Figure A-4. Figure A-5 shows the keyboard connector whose only changes are the signals on pins 15 and 19. Figure A-6 shows the new disk-eject switch connector. Figure A-8 shows the new expansion connector, J14. Figure A-9 shows the new internal modem connector.

Keyboard

The IIc Plus utilizes the new Apple Standard Keyboard, less the numeric keypad. Note that the **Closed Apple** key has been labled as the **Option** key (See Figure A-7).

Accelerator Circuitry

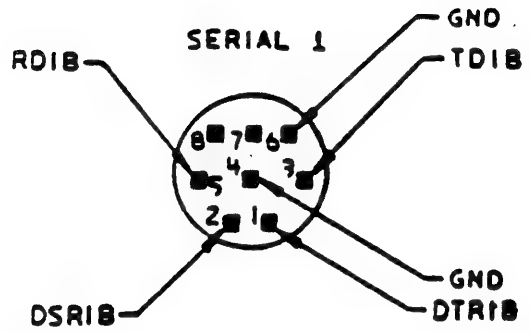
The accelerator circuitry consists of an 84 pin PLCC custom gate array referred to as the accelerator, a 16 MHz crystal oscillator, and two 8Kx8 static RAM's that are used for data caching. The operation of the accelerator circuitry is described in Part 2 of these preliminary notes.

Minor Circuitry Changes

There is some additional new circuitry on the Apple IIc logic board. The sound hybrid chip has been replaced with the equivalent discrete circuit. Several transistors have been added to enable push-button electric eject from the internal drive and the "disk activity" LED. An LS125, LS08, LS11 and 555 timer replaces the LS14, CR2 and CR3 to assist in the "glue logic" for the 3.5" drives. The 555 timer allows the internal 3.5" drive to remain enabled for up to a second after the last access to it, unless another enable or reset is asserted. Finally, eight 47 ohm resistors replace the filter between the IWM and external drive connector.

Appendix A
Connector Pinouts and Keyboard Layout

Figure A-1, IIc Plus Serial Mini-DIN Connectors (J6 and J12)



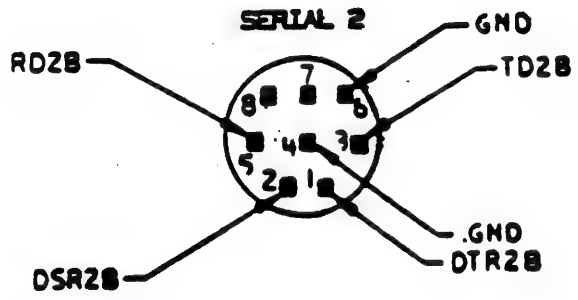


Figure A-2, IIc Plus External Disk Drive Connector (J2)

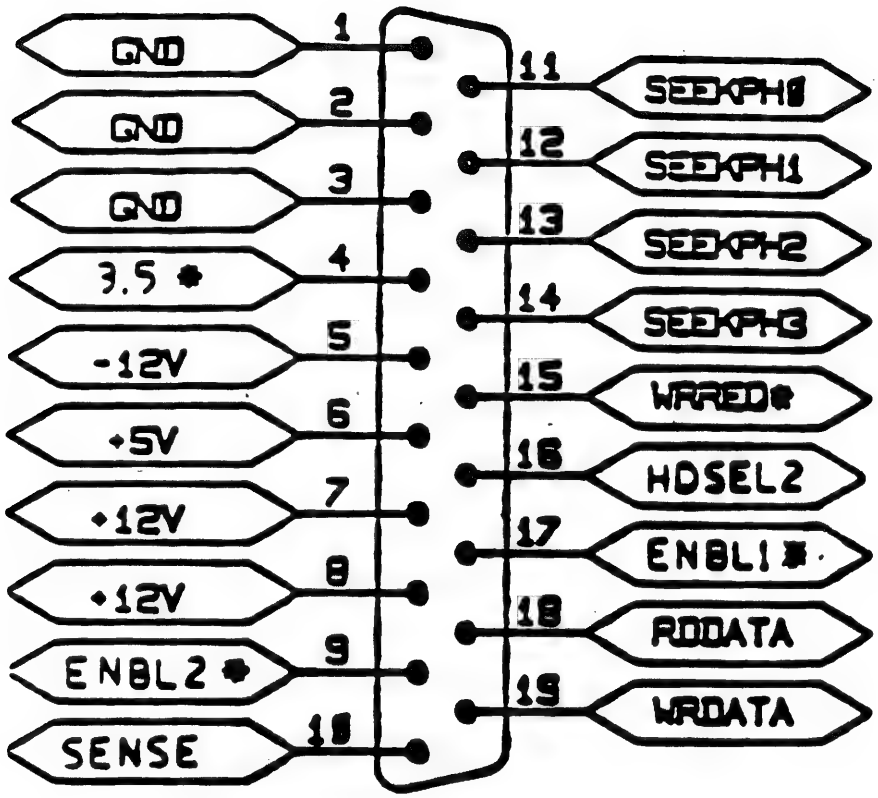


Figure A-3, IIc Plus Internal Disk Drive Connector (J8)

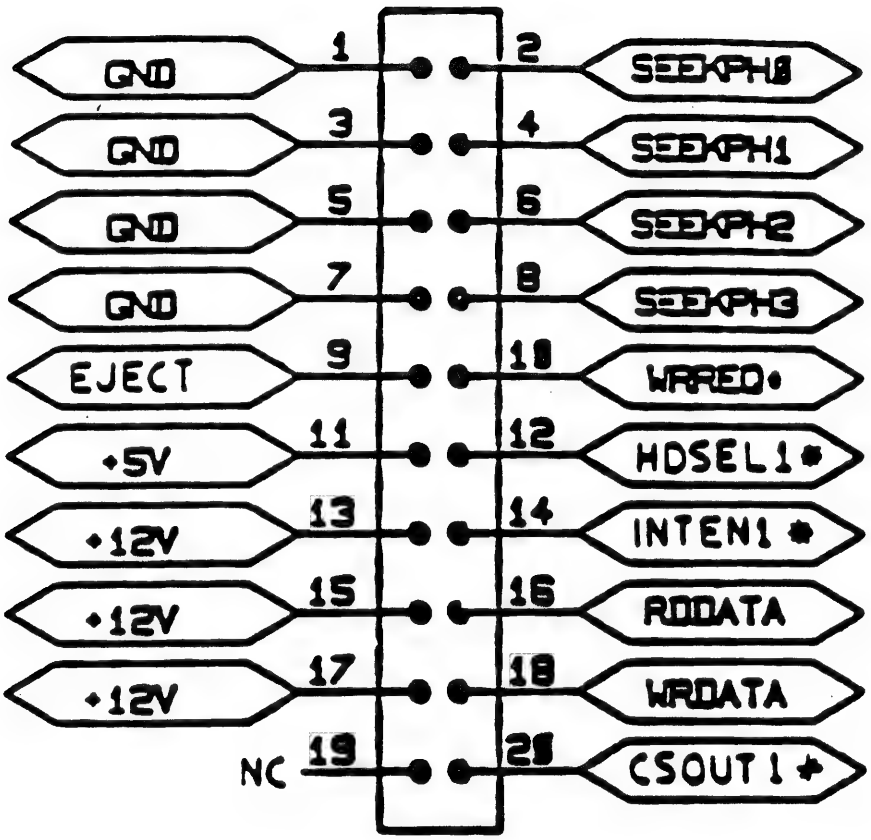


Figure A-4, IIc Plus Power Supply Connector (Internal – J1)

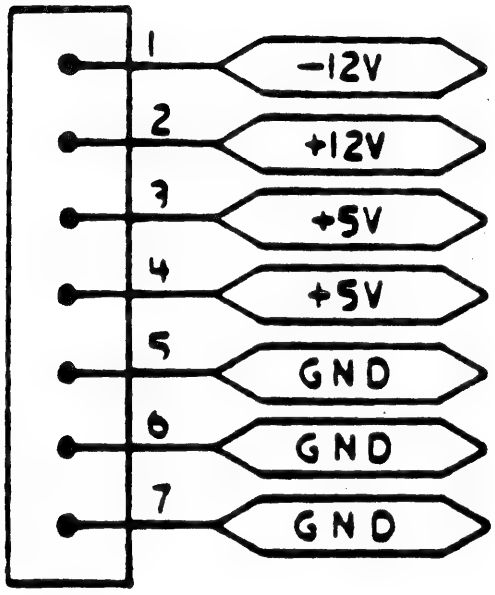


Figure A-5, IIc Plus Keyboard Connector (Internal – J9)

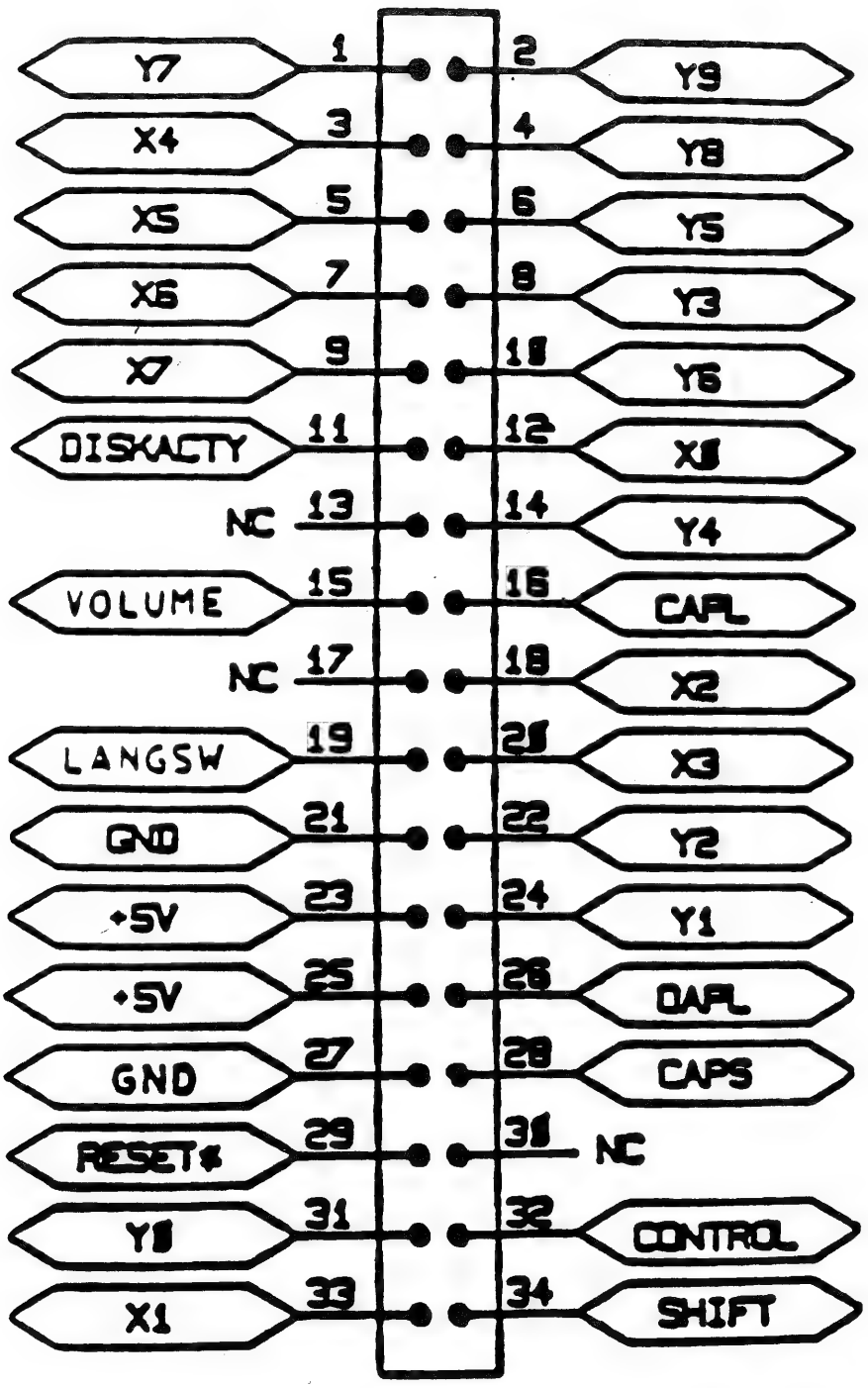


Figure A-6, IIç Plus Disk Eject Switch Connector (Internal – J7)

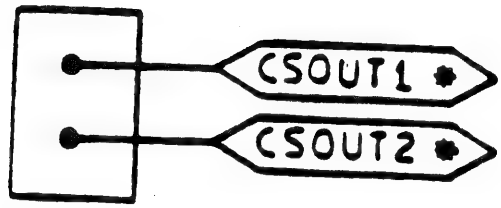


Figure A-7, IIc Plus Keyboard Layout

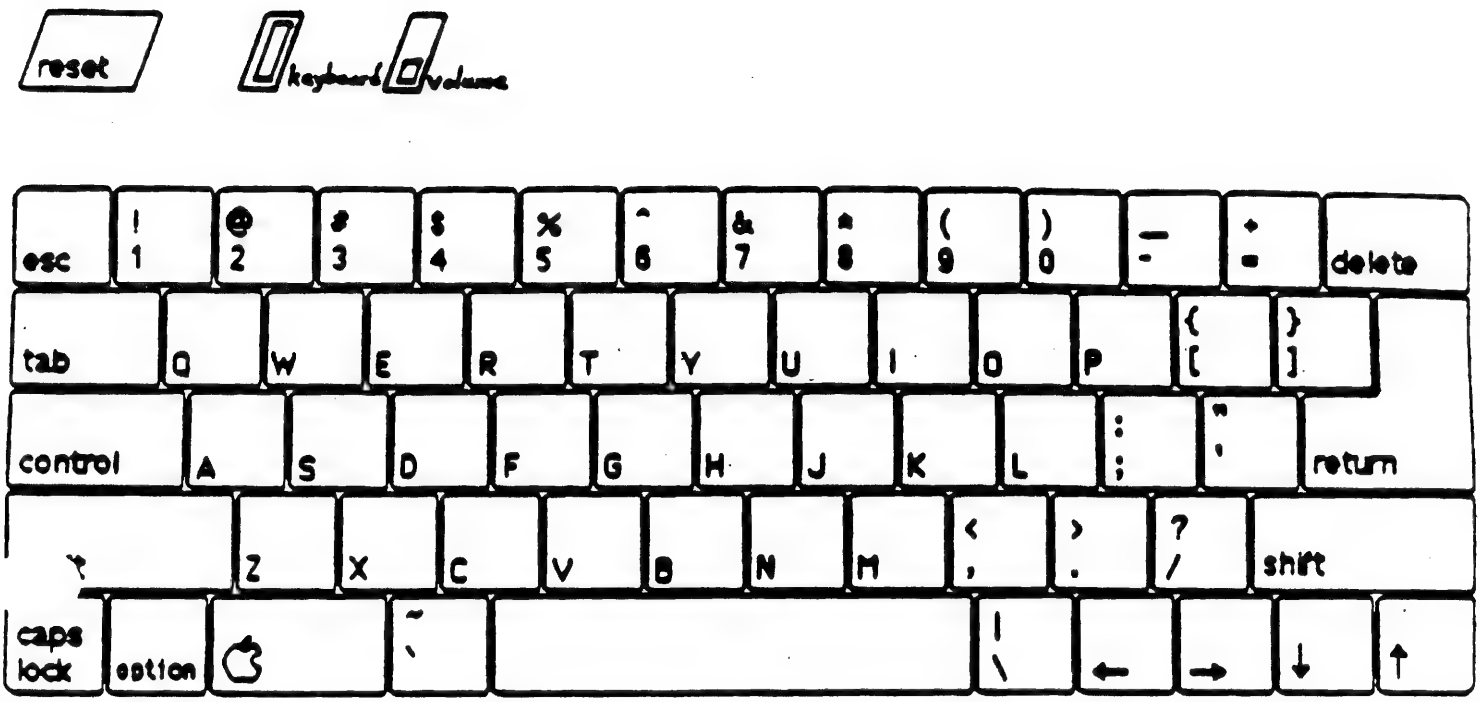


Figure A-8, IIc Plus Expansion Connector (Internal – J14)

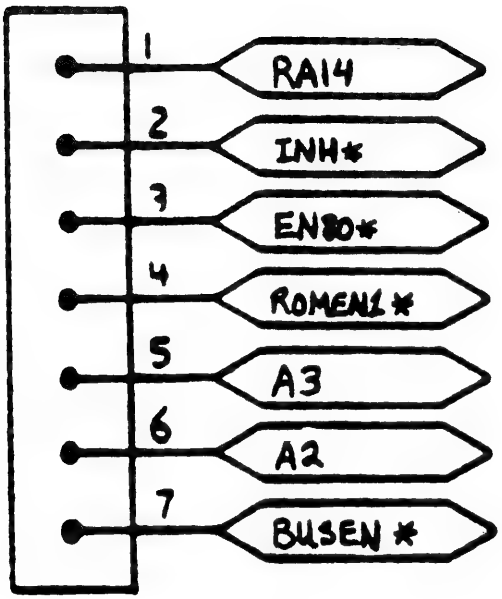
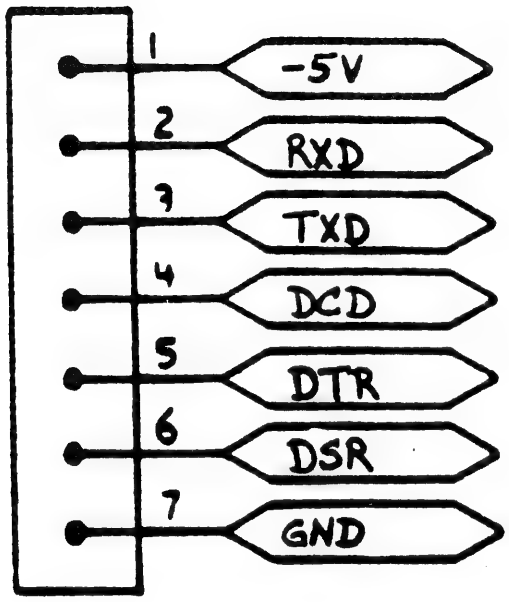


Figure A-9, IIc Plus Internal Modem Connector (J15)



Part 2

The Accelerator Feature

Introduction

The accelerator is a speed controller and a RAM data cache for the IIc Plus. When it is active, the accelerator circuitry controls the speed at which the IIc Plus runs and also caches data the processor has run through for possible future use. The accelerator uses 8K of RAM for the cache. When data comes from the cache, the processor is running at 4 MHz. Applications using the accelerator will obtain a performance improvement which will vary depending on the application.

Fast Versus Synchronous Mode

When you power up the IIc Plus, it will automatically be running in 4 MHz mode (referred to as fast mode). Because parts of the firmware and some software applications have timing critical code which must be run at 1.024 MHz clock speeds to function properly, there is also a 1.024 MHz mode (referred to as synchronous mode) available. During the startup process, the system sets which "slots" will run fast and which will run synchronous.

Synchronous mode

Slots 1,2,5 and 6 are hard wired to run in synchronous mode. These slots are used for printers, modems and disk drives and all have timing critical code which must run in synchronous mode to function properly. In addition, the speaker and the game port are also set to run in synchronous mode. Joysticks and paddles will not function in fast mode and the speaker is unusable in fast mode. To obtain synchronous mode, the hardware traps accesses to $C0Nx where N is the slot number plus 8. This causes the system to sync up with the 1.024 MHz Apple clock for 50 ms. Repeated accesses to these locations will continue to trigger the 50 ms timer. This only occurs for slots 1,2,5 and 6. When $C03x or $C070 is accessed, the system goes into synchronous mode for 5 ms. This allows the hardware to work properly with its timing critical code in these areas. Any other access to Apple memory will be synchronous for that read or write access cycle only.

It is also possible for the user to force the system to always run in synchronous mode. This is done on power-up or during Control–Open Apple–Reset by holding down the Escape key **before** booting. If the machine is in complete synchronous mode, the word "Normal" will appear on the booting splash screen or blanked screen just before the logo. If the word "Normal" does not appear, the system is still running in fast mode. It is essential that the Escape key be held down first or there is no guarantee that the system will boot in normal mode. Once the system is booted into the synchronous mode, the machine must be cold booted or restarted to put the machine back into its default fast mode. This is accomplished by re-booting without the Escape key down.

Fast Mode

The remaining slots; 3,4,7 are set to run in fast mode. This allows the 80 column firmware, the mouse and the RAM expansion card data to be cached and thus run at at a speed of up to 4 MHz. The ROM is also executed in fast mode unless it makes accesses to any of the hard wired synchronous slots. As mentioned above, any read or write access to Apple memory will be synchronous for that access cycle only, even in fast mode. All other accesses to Apple memory will be synchronous (for one access cycle) only if the read data is not available in the cache or if a write to the Apple memory is performed. The increase in

performance actually occurs during the processor reads when the data is available in the cache. This occurs quite often.

What Will and What Won't Work

There are some software applications that will not function correctly with the accelerator. This will occur in applications that use DMA or other timing critical code that must run in the synchronous mode to operate properly. There will also be some hardware products that will not function for the same reasons. Some software products -- especially games -- will function in fast but will be too fast to operate. If this is the case, the program should be booted in synchronous mode. Unfortunately it is a trial and error process to determine what will not run in fast mode. The positive side to this is that most applications and hardware products should work with no problems. This is not true for any hardware or software products that use DMA. DMA is not supported on the accelerator and will not work in fast mode.

Enabling the Accelerator

The accelerator is automatically enabled (fast mode) when power is initially applied to the computer. Once the accelerator has been disabled, only the user can re-enable it. You can re-enable the accelerator by performing a normal (without the Escape key held down) cold start as follows:

* Turn the power off (wait 10 to 15 seconds) and then back on (cold start), or
* With the power on, press Control–Open Apple–Reset (release Reset before releasing Control–Open Apple keys)

By performing one of the above steps, the accelerator will automatically be enabled regardless of the past state of the accelerator.

Disabling the Accelerator

If the user disables the accelerator, the computer is forced to run in the synchronous (1.024 MHz) mode. You can disable the accelerator by performing a "cold start" as follows:

* If the computer is not powered up, hold down the Escape key and set the power switch to on
* If the computer is running, hold down the Escape key and press Control–Open Apple–Reset (to ensure that the accelerator is disabled, release the keys in the following sequence: first the Reset key, next the Escape key, and finally the Control–Open Apple keys)

Remember! This will only work if you hold down the Escape key **before** you attempt to "cold start" the machine. If you have successfully disabled the accelerator, the word "Normal" will appear at the top of the screen just under the title. If the Escape key has been held down, and that message does not appear on the screen, the user should repeat the above cold start procedure.

Programming Hints

The introduction of an accelerator into the IIc Plus will require the developer to take a different mind set when programming. There are some built in features that can be used by developers to make their programs compatible with future versions of any machine.

The ROM WAIT Routine

Apple will guarantee that the WAIT routine at $FCA8 in the ROM will run in synchronous mode to preserve timing loops. This is true no matter if the machine was booted in fast mode or in synchronous mode. The machine will continue to run in synchronous mode for up to 50 ms after exiting the wait routine. To insure the timing, the WAIT routine has been thoroughly rewritten. It is important that developers not use any other entry point except $FCA8 because $FCA8 is guaranteed to still be there in the future. Also note that there is no way of knowing what speed the machine will exit the WAIT routine in, except that within 50 ms it will return to fast mode if that is how the machine was booted. If the machine was booted in synchronous mode, it will exit the WAIT routine in synchronous mode and will continue that way. Remember that the WAIT routine is a **minimum** wait and should **never** be used as an absolute time. Apple will not support developers whom accidently enter the WAIT routine from the upper half of the ROM. Any application that does this **will** crash!!

Booting into Escape Mode

Many programs use the Escape key as a main menu item. In the past there has been no problem with this because no one ever used the Escape key. Because the accelerator uses the Escape key to boot into synchronous mode, some programs will interpret this as a menu choice. The simplest way to correct this problem is to clear the keyboard strobe when booting an application. If an application looks for a specific key on boot up, it should clear the keyboard strobe first and then look for the required key.

Temporarily Disabling the Accelerator

If a developer needs to temporarily disable the accelerator to insure that a section of code runs in synchronous mode, there are some soft switches that can be used. Any access to $C09A or $C0AA will cause a 50 ms period of synchronous operation. These are ASIC command registers for serial ports 1 and 2. Accesses to these switches will not affect the operation of the program. If a shorter delay is required, $C070 can be used for a 5 ms delay. This should only be used in applications that do not use the paddles because $C070 will cause them to be reset.

Firmware Identifiction Bytes

Three ID bytes in ROM at locations $FBB3, $FBC0, and $FBBF allow you to differentiate between the previous (memory expansion) version of the Apple IIc and the IIc Plus. The values of these bytes are $06, $00, and $03, respectively for the memory expansion Apple IIc and $06, $00, and $05, respectively for the IIc Plus.

Part 3

IIc Plus Firmware and 3.5" Drive Support

Overview

Goals

The purpose of the IIc Plus Project is to produce a faster, cost reduced version of the Apple IIc with a built-in 3.5" disk drive. Internal to Apple, the 3.5" drive has been referred to as the "Unified" Sony drive because it is compatible with both Apple II and Macintosh computer systems. The formal name of the drive is the Apple 3.5 Drive.

This document describes the firmware functions for the IIc Plus. Areas where the firmware is different (or operates differently) in the IIc Plus than previous versions of the Apple IIc are organized with section titles that refer to the respective products.

Remember! The IIc Plus system is running much faster due to the introduction of the accelerator chip. See Part 2 of these preliminary notes for details on the new accelerator feature.

Hardware Requirements

The first requirement is the availability of a RAM buffer of at least 800 bytes. Due to the 500 KHz data rate (16 uS per Byte) of the 3.5" disk drive and the extreme rarity of *safe* RAM in Apple II systems, it is necessary to implement a RAM buffer *outside* the normal RAM address spaces of an Apple II. A custom gate array chip provides the general interface ("glue logic") for implementation of a RAM buffer on the Apple IIc. This chip is referred to as the MIG (Magic Interface Glue) chip. At least 8K of ROM is also needed for drivers to control the three types of disk drives.

Last, though critically important, the 65C02 must execute at the standard Apple 1.024 Mhz rate. Certain portions of the firmware are dependent on reliable execution timing, and will fail to function at speed variances of more than plus or minus 10%.

Assumptions

The IIc Plus firmware assumes the MIG implementation, especially in the area of the way the RAM buffer functions. Briefly: the RAM is addressed as multiple $20 byte "pages", with an auto-increment (to next page) to locations at RAM buffer address+$20 through +$3F. Note that this RAM space is for internal use only and is not accessible to developers or users.

The presence of a standard IWM chip is also presumed. The behavior of the IWM, with respect to timing in particular, is heavily depended upon. However, future enhanced versions of the IWM can be expected to function with the IIc Plus firmware if they pass the more rigorous test of compatibility with existing Disk II copy-protection schemes.

With respect to Disk II's, it is assumed that either calls will be made through the standard entry points to perform disk operations or that programs will directly access the IWM to control the drive. Programs such as "Uni-DOS 3.3" that use both methods will **not** function. However, programs may be made to operate in both modes providing that they set up the IWM in the desired mode before attempting direct read to or write from the disk drive.

Configurations

The IIc Plus version of the firmware supports up to six disk drives (including the internal). The hardware is designed to support only five drives. (See the Part 1 of this preliminary note for the IIc Plus hardware information.) The power supply in the IIc Plus can only support a maximum of four drives (one internal and three external). For these reasons, Apple recommends that only three external drives be attached to the IIc Plus (for a total of four drives).

Driver entry points correspond to "slots" 5 ($C5nn) and 6 ($C6nn). Calls made to the entry points in $C5nn apply to the built-in 3.5" disk drive as well as up to three (Under ProDOS 8 or Pascal 1.3) external 3.5" drives . Apple recommends using Smartport calls to talk to the 3.5" drives. Smartport calls are supported @ ProDOS entry+3. Calls to the $C6nn entry points are applied exclusively to up to two external Disk II's (5.25"drives).

The built-in drive is a "bare" Sony drive, lacking the daisy-chain gate array of a standard 3.5" drive. It is always identified as "slot 5, drive 1", device number $5n, and unit 1 of the Smartport devices. The first external 3.5" drive is drive 2. Additional 3.5" drives are mapped to slot 2 by ProDOS 8.

Disk II's on the chain are always accessed as slot 6. Whether or not Disk II's are present on the chain, two drives will always be reported as present. This is due to the age old problem of inadequate ways to test for the actual presence of a drive. In any case, the firmware will report "No Device" error whenever there is no drive or no media mounted in the drive. It is much better to assume drives, rather than make the drives inaccessible simply because there was no media mounted when the system was booted.

Daisy Chain Configurations on the IIc Plus (with respect to firmware requirements)

Built-in only

Built-in + Apple 3.5 Drive (up to 2)

Built-in + Apple 3.5 Drive (up to 2) + Disk II (up to 2)

Built-in + Apple 3.5 Drive (up to 2) + UniDisk 3.5 Drive

Built-in + Apple 3.5 Drive (up to 2) + UniDisk 3.5 Drive + Disk II (up to 2)
(maximum total of 5 drives)

Built-in + Apple 3.5 Drive + UniDisk 3.5 Drive (up to 2)

Built-in + Apple 3.5 Drive + UniDisk 3.5 Drive (up to 2) + Disk II (up to 2)
(maximum total of 5 drives)

Built-in + UniDisk 3.5 Drive (up to 3)

Built-in + UniDisk 3.5 Drive (up to 3) + Disk II (up to 2)
(maximum total of 5 drives)

Built-in + Disk II (up to 2)[1]

[1] Note that these configurations are based on the maximum allowances for the hardware, but not the power supply. While these should work, Apple does not advise overloading the power supply by adding an additional drive.

Startup (Boot) Sequence

When the IIc Plus is powered on, prior to execution of the boot routines, the initialization routines in the firmware are executed. During initialization the chain is initialized, starting with the UniDisk 3.5 Drives and working backwards to the first Apple 3.5 Drive in the chain. At this time any and all "shipper disks" (as well as completely blank diskettes) are ejected. A table is created in the RAM buffer that indicates how many drives are attached (excluding Disk II's) and what type of drive each is (UniDisk 3.5 Drive or Apple 3.5 Drive). It is assumed that there are two Disk II's attached; because the existence of Disk II's can not be tested.

When boot code is called (via entry at $C500), the internal drive is accessed to read block 0 into $800. Locations $800 and $801 are tested for a $01 and non-zero respectively, and the code starting at $801 is executed. If the diskette is un-readable (or the locations tested are not as expected) and it is not an auto-boot, the message is displayed "Unable to find a bootable disk online."

If the internal drive cannot be read, and it is an auto-boot sequence (locations 0 and 1 contain $00 and $C5 respectively), the **first** 3.5" external drive is read. If it cannot be read, a Disk II boot is initiated.

The Disk II boot is somewhat different than what has been done in the past. Block 0 is read into $800 using the ROM routines, but no check for a valid read is made. The IWM is then set into vanilla Disk II mode. Then track 0, sector 0 is read again into $800 using "traditional" Disk II boot code. If the code read meets criteria ($800=01 & $801>00) the code starting at $801 is executed. The reasons for this double read is more thoroughly examined under "Disk II and Non-ProDOS Applications" later in this document.

ProDOS and Smartport Calls

Standard ProDOS Support

The standard four ProDOS calls (Status, Read, Write, and Format) are fully supported for all three types of drives that are controlled by the IIc Plus firmware.

Interrupts are disabled during all ProDOS calls. This does not affect the operation of interrupt driven devices -- such as mice. The only noticeable difference is that a mouse will not move on the screen while the disk is being accessed. The UniDisk 3.5 Drive routines shut down interrupts for the duration of each call request (up to 8 seconds).

Device size is returned from the status call in the X and Y registers. It is important that no assumptions be made about device size since Disk II's, Apple 3.5 Drives, and Smartport Bus devices such as the UniDisk 3.5 Drive can all be attached to the same card.

Standard error codes ($27=I/O error, $28=No Device, $2B=Write Protected, $2D=Invalid Block Number, $2F=No Media) are returned as appropriate.

Smartport Support

All (non-extended) Smartport calls for block devices are supported. In addition, some generic device specific calls are also implemented.

The calls supported are listed below. Those marked with an asterisk (*) are not available when accessing a Disk II.

Call #	Name	Comments
$00	Status	Codes $00 and $03 (DIB) only.
		Includes Unit $00 specification
		Interrupt Status always = $00
$01	Read Block	Standard 512 byte Read
$02	Write Block	Standard 512 byte Write
$03	Format	
$04	Control	Only control codes $00 (Reset device)
		and $04 (Eject) are supported. Unit
		$00 calls NOT supported (see below).
$05	Init	Unit $00 only.
$08	Read*	Byte count = 524 ($20C) only.
$09	Write*	Byte count = 524 ($20C) only.

Calls $08 and $09 are not supported by the Disk II code because the standard media format does not allow for variance in sector size. These calls will return error code $21 (BADCTL) when referencing a Disk II type device.

Unit numbers for the Smartport devices are mapped according to the order that the devices appear in the chain. Thus Apple 3.5 Drives will always have lower Unit numbers than UniDisk 3.5 Drives in the same chain.

Control calls with a unit number of $00 are not supported. On the Apple IIc these were used to enable (allow) interrupts from the disk port. The error code $21 (BADCTL) is returned when any unit $00 control call is issued.

All other standard error codes are supported. Device specific errors from Smartport Bus devices are also returned.

Device specific calls for the UniDisk 3.5 Drive are supported as they were on the UniDisk 3.5 Drive controller card code. These calls are to be considered specific to the UniDisk 3.5 Drive, and no attempt has been made to emulate them for the Apple 3.5 Drive (see "Copy Protection on Apple 3.5 Drive" later in this section).

Device specific calls for the Apple 3.5 Drive that are available in the Apple IIGS (built-in controller) are not available in the IIc Plus firmware. This is due primarily to processor speed considerations on a pre-GS Apple II (again, see "Copy Protection on Apple 3.5 Drive" later in this section). A secondary constraint is the necessity of bank-switching code space in both firmware implementations, making the idea of "hooks" nearly impossible.

Special Considerations

Disk II and Non-ProDOS Applications

When the primary drive (for example, Drive 1) of the boot device is a Disk II, a unique situation arises. The IIc Plus firmware tells ProDOS (and other languages that may use the firmware protocol) to use the ROM driver rather than the RAM based code. And because the firmware must support multi-device configurations, each with a different IWM mode, firmware for Disk II is a necessity. The difficulty is that boot behavior is different between ProDOS devices and Disk II's.

The Disk II boot code, written in nearly 10 years ago, reads one 256 byte sector into $800, tests that location $800 contains a $01, then jumps to the code at $801. Because of code space limitations (256 bytes of ROM), a bad sector 0, as is the case in many copy-protected disks, can be successfully loaded and executed. Such valid bad sectors cannot be read by standard driver routines.

A second category of Disk II applications are those that expect to control the Disk II with RAM based routines. DOS 3.3 and early versions of Pascal are such applications. The ID bytes must be those of a Disk II, or the application will not access the device. The solution, on first glance, is to simply use the old Disk II boot code to boot. But that is only a partial solution.

The standard ProDOS loader code (block 0) is somewhat adaptive. If the boot device is a Disk II (old boot code) it assumes only 256 bytes have been read. It uses the boot ROM to read in the other half of the loader code and *creates* a Disk II block read routine in RAM from code in the boot ROM. If, however, the boot device is a ProDOS block device, the loader assumes that the entire 512 byte block 0 has already been read. The ProDOS loader assumes that a device is a Disk II or it's a block device, but not that it can be both!

The solution is to actually boot using both methods. First block 0 is read into $800. Then the old Disk II boot code is executed to read sector 0 on track 0. In the event that it's ProDOS, the entire block is there for the ProDOS loader. If it's not ProDOS, then even if the firmware drive could not read block 0, the loader code in sector 0 can treat the controller as it would an old Disk II card. And everybody is happy.

Differences Between the IIc Plus and previous Apple IIc's

The IIc Plus firmware has to support, in addition to the standard external devices, a naked Sony 3.5" internal drive. The primary implementation difference is in the use of the IWM time-out mode for this and no other 3.5" drive (also see "IWM Reset on the IIc Plus" later in this section). The MIG chip implements a special "internal" mode to allow double-duty for one of the IWM's drive select lines. So the internal drive requires a different set-up than its external cousins, the Apple 3.5 Drive. The previous Apple IIc's have none of this special set-up code.

In previous models of the Apple IIc, most -- if not all -- of the upper half of the ROM was blank and unused. Apple used the outdated cassette port ($C02X) soft switch to toggle between the two banks of ROM. Developers who used this switch for purposes other than its intended use (for example, to redirect the sound output when sound is not desired) can accidently end up in the wrong half of the ROM. In the earlier versions of the IIc, this did not present a major problem because the firmware automatically put the application back into the low half of the ROM. In the IIc Plus however, there is no empty space on the

upper half of the ROM. Because this empty space was used to hold disk code, any program that unintentionally gets into the upper half of the ROM will suffer disastrous consequences.

The state of the language card RAM must be preserved on entry to the IIc Plus drivers, then re-enabled whenever access may be needed by the code. This adds some time overhead between subroutines on the IIc Plus implementation.

The IIc Plus contains an accelerator chip which can increase the system speed up to 4 MHz. For a detailed description of the accelerator chip, refer to Part 2 of these preliminary notes.

IWM Reset on the IIc Plus

The internal 3.5" drive and the Disk II's that may be attached to the IIc Plus use the IWM motor on time-out feature. When switching from one type of device to another it is necessary to wait until the motor has timed out in order to change the IWM mode for the new device. A guaranteed means of shortening the time-out period (to zero) is to reset the IWM.

On the IIc Plus, a software controlled reset of the IWM is implemented using the bank select feature of the MIG chip. These outputs on the MIG chip are used for "internal" mode drive selection, instead of bank select.

The firmware resets the IWM whenever a mode change is desirable and either motor on enable is true. However, if for any reason the reset cannot be implemented, the code does not rely on this method of mode switching. If the reset feature is removed, no changes in the code are necessary. It will simply result in some delay when switching between unlike devices.

Limitations

Smartport Extensions

The extended Smartport calls are NOT supported by the IIc Plus firmware. The extended calls were invented with only the Apple IIGS in mind. Increasing the ROM allocation from 8K to 16K would allow for these calls to be supported in GS and future GS style systems.

Likewise, device specific calls for the Apple 3.5 Drive only took into consideration the Apple IIGS environment. The speed and architecture of pre-GS machines are unable to support all but the simplest of device specific calls: Eject. Again, a larger ROM would make these features available to GS class machines. But more ROM will not bring these features back to older Apple II systems.

Copy Protection on Apple 3.5 Drive

The IIc Plus firmware does not provide support for program copy protection on the Apple 3.5 Drive. The firmware is designed to work only with a 1 MHz processor speed, so as to be executable by all machines in the Apple II line. It is due to this limitation that copy protection methods employed on a native-mode 2.7 MHz Apple IIGS cannot be implemented.

A method can be devised to support copy protection, but it would not be compatible with either the Apple 3.5 Drive on the IIGS or with methods employed with the UniDisk 3.5

Drive. Any encryption method that would function across the product line would likely be very easy to break in the upscale machine(s).

Direct Access to Disk II's

Many existing games and applications that are sold in Apple 5.25" format expect to directly control the Disk II attached to the controller. Only a few expect to directly control the Disk II, but call the ROM driver for anything else. Examples are: UniDOS 3.3 and Dazzle Draw.

The problem arises (and makes such programs incompatible with the IIc Plus implementation) after accessing any of the drives via the ROM driver. The direct access of a Disk II is no longer possible because the IWM is left in a non-Disk II state. Even the firmware driver for Disk II leaves the IWM in "async" mode.

Copy protected programs that follow all the ROM ID conventions may not even recognize the controller as a Disk II, simply because it has ProDOS device IDs as well. Such programs may not boot at all.

In summary: Applications should either use the ROM drivers, or not. But they should not do both direct access and ROM driver calls. If they must do both, they must be responsible to set-up the IWM before direct accesses are made.

(Future) Higher Data Rate Drives

It is unlikely that the current hardware implementations, even with a new higher rate IWM, can be made to directly control higher data rate disk drives. Code changes are even less likely because the 500 KHz rate pushes a 1 MHz 6502 nearly to its limits.

A higher data rate drive can, however, be made to function with the existing hardware and firmware if it is implemented as Smartport Bus device. The Smartport Bus will be in danger of being neglected when UniDisk 3.5 Drives are phased out. It should not be abandoned, unless the IWM itself is abandoned.

Addendum

IIc Plus Preliminary Reference Addendum

Section 1: Dealing with Reset

When handling RESET, the IIc Plus machine decides to disable or enable system speed acceleration.

As documented is elsewhere, the 65C02 microprocessor jumps through a vector at $FFFC when it gets a RESET. This vector, in all Apple II ROMs, points to $FA62, where the ROM reset handling routine resides. After taking care of necessary business, the ROM reset handling routine jumps through the user reset vector at $3F2, after first making sure the power-up byte (at $3F4) is valid.

Adam Ant's built-in reset handler at $FA62 first looks at the escape key, and disables system speed acceleration (normal mode) if it is depressed, and enables system speed acceleration (accelerated mode*) if it is not. It does this **before** it ever looks to see if any modifier keys (like the Apple key or the command key) are depressed. This fact has the following implications:

1. Pressing control-reset will change the state of the machine. If the user had booted into normal mode, but presses control-reset, the IIc Plus will go into accelerated mode. The reverse is true as well – normal mode will be enabled if the user presses control-esc-reset (for any reason), even if the machine was in accelerated mode. Enabling normal mode via control-esc-reset puts the word "Normal" on the 40-column text screen, just as it does during the boot process.

2. The above will be true with your reset handler provided you have not replaced the system reset vector at $FA62. If you have replaced this vector, the machine will **always** come up in accelerated mode, regardless of the disposition of the escape key. As is stated elsewhere, **never** steal the system reset vector.

* When user's boot preference selects accelerated mode the system can accelerate to 4MHZ. The system will, however, still operate at the 1.023 MHZ speed during certain operations. In accelerated *mode*, the processor may be said to be operating at fast *speed* (4MHZ) or synchronous *speed* (1.023 MHZ).

Section 2

This section left empty.

IIc Plus Preliminary Reference Addendum

Section 3: IIc Plus Smartport device identification

On the IIc Plus, the Device Information Block returned by a Smartport $03 status call contains **the same device subtype value ($0) for both Unidisk 3.5 and Apple 3.5 drives.** Programs using the device subtype value to **differentiate between the two drives** should further identify the device.

If the device subtype is $0 indicating a Unidisk 3.5, then also issue a Smartport $05 Unidisk 3.5 STATUS call. An Apple 3.5 drive will then be indicated if the Smartport $05 status call fails and returns the carry bit set, since this call is a Unidisk 3.5 specific call. Note the IIc Plus does not support Smartport calls specific to the Apple 3.5 disk drive.

Section 4: Mouse/VBL Handling & Accelerated Serial I/O

Using the mouse firmware in any mode except transparent mode on Adam Ant will cause the system to run at 1.023 MHz, even when the system is set to run fast. For applications that encounter this problem, the following two solutions that are recommended. Applications should, upon return from the example code routines that follow, do any banking in of RAM they require.

Using the mouse firmware in any mode except transparent mode on Adam Ant will cause the system to run at 1.023 MHz, even when the system is set to run fast. For compatibility reasons the serial ports 1 and 2 were set to run at 1.023 Mhz (synchronous speed) and accessing the status registers starts a 50ms slow down timer. During VBL and mouse interrupt handling, the system checks the serial ports to see if there is an interrupt pending, accessing a status register and starting a 50ms slow down timer. Since mouse interrupts are handled when VBL occurs and 50ms is 3 times as long as the VBL period, the system is forced to almost always be running at 1.023 MHz. For applications that encounter this problem, there are two solutions that are recommended:

The best solution to the problem, for mouse handling, is to simply operate the mouse in transparent mode. This way the mouse handler will not check the serial ports and will not activate the slow down timer. This is recommended as the cleanest solution.

The second solution involves subroutine calls at the beginning and the end of the application. This temporarily forces ports 1 and 2 to run fast so that accessing them during interrupts will not slow down the system. The following conditions must be met before using this in a program:

1) The application must not depend on the timing of the `WAIT` routine ($FCA8) because the patch will also force the `WAIT` routine to run at fast speed. (This can be compensated for by replacing calls to the $FCA8 `WAIT` routine with calls to the `WAIT` routine listed below.)
2) Applications that write directly to the serial ports will need to make sure that there is no timing critical code. Most application code writing directly to the serial ports will not be timing critical. In the instance of an application containing timing critical code that writes directly to the serial ports, this code may be made to run at the synchronous 1 MHz speed. This is done by accessing a softswitch, such as in the revised `WAIT` routine below, that triggers a slow down timer.

3) In order to call the 233ms break ROM routine and obtain an accurate 233ms timing, a program would first have to deactivate speed-up of ports 1 and 2 and then reactivate speed up of these ports upon return from the 233 ms break ROM routine.

Note the screen image may be momentarily disrupted when these routines are called due to accesses to screen setting softswitch locations.

This part should be called at the very beginning of the application before anything else is done:

```
                X6502                   ;enable EDASM 65C02 instructions
accel.call      equ     $C7C7           ;rom entry point
ROMIN           equ     $C082           ;switches in the ROM
notadam         equ     $01
adamant         equ     $82
RdDHIRES        equ     $C07F
DHIRESOn        equ     $C05E
DHIRESOff       equ     $C05F
Rd80Col         equ     $C01F
On.80Col        equ     $C00D
Off.80Col       equ     $C00C
speed12         jsr     sysid           ;JSR to this location
                bne     speedret        ;branch if system not (adam ant & $fbbf = 5)
                jsr     testaccel       ;has user disabled acceleration?
                bne     speedret        ;yes- forego speed up of slots 1,2
                jsr     SaveDHIRES      ;preserve dbl hi-res, 80 Col setting
                lda     #$00            ;speed up ports 1 &2
                pha
                lda     #$61
                pha
                lda     #$06
                pha
                jsr     accel.call
                jsr     UnsaveDHIRES    ;restore dbl hi-res, 80 Col setting
speedret        rts                     ;back to users routine
```

This part must be called at the very end of the application before it quits:

```
synch12         jsr     sysid           ;check if system is adam ant
                bne     synchret        ;branch if system not (adam ant & $fbbf = 5)
                jsr     testaccel       ;has user disabled acceleration?
                bne     synchret        ;yes- forego reset of slot 1,2 speed
                jsr     SaveDHIRES      ;preserve dbl hi-res, 80 Col setting
                lda     #$00            ;slow down ports 1 &2
                pha
                lda     #$67
                pha
                lda     #$06
                pha
                jsr     accel.call
                jsr     UnsaveDHIRES    ;restore dbl hi-res, 80 Col setting
synchret        rts                     ;back to users routine
speedbuff       dw      0
testaccel       lda     #<speedbuff     ;push high byte
                pha
                lda     #>speedbuff     ;push low byte
                pha
                lda     #5
```

```
                pha
                jsr     accel.call
                lda     speedbuff+1
                and     #8
                rts

TempDHRes       dfb     0
Temp80Col       dfb     0
SaveDHIRES      lda     RdDHiRes        ;save dbl hi-res setting
                sta     TempDHRes
                lda     Rd80Col         ;and 80 Col setting
                sta     Temp80Col
                sta     Off.80Col       ;turn 80Col off before calling ROM
                rts                     ;setting saved, return

UnsaveDHIRES    bit     TempDHRes       ;was DHIRES off?
                bmi     DHIOff          ;yes- branch
                sta     DHIRESOn        ;no- turn DHIRES on
                bpl     Unsave80Col     ;branch unconditionally
DHIOff          sta     DHIRESOff       ;turn DHIRES off
Unsave80Col     bit     Temp80Col       ;was 80Col on?
                bpl     unsaveret       ;no- branch to rts, 80Col already off
                sta     On.80Col        ;turn 80Col on
unsaveret       rts

idloc1          equ     $FBB3
idloc2          equ     $FBC0
idloc3          equ     $FBBF
idval1          equ     $06
idval2          equ     $00
idval3          equ     $05
sysid           lda     ROMIN           ;ROM must be enabled
                ldx     idloc1          ;check id bytes
                cpx     #idval1
                bne     idret
                ldx     idloc2
                cpx     #idval2
                bne     idret
                ldx     idloc3
                cpx     #idval3         ;found all the bytes? then z flag =1,else z=0
idret           rts
```

```
kbd             equ     $C000
newwait         phy                     ;wait routine
                ldy     #$D0            ;$C0D0 guaranteed 50ms slow down on
                phx                     ;adam ant
                sec
                tax
nw1             equ     *               ;first pass may be accelerated until
                lda     kbd,y           ;<-this instruction starts slow down
                txa                     ;min delay = 1/2(50+25A+5A^2) cycles
nw2             equ     *               ;min delay = (A - 12) fewer cycles
                sbc     #$01            ;than standard wait
                bne     nw2             ;see Apple II Misc. Tech Note #12
                dex                     ;for further details on wait routine
                bne     nw1
                plx
                ply
                rts
```

Section 5: Handling of Time-dependent Routines in Accelerated Mode

---

System acceleration may cause time-dependent routines such as joystick reading to malfunction, if not taken into account.

Waiting routines, such as those many programs put around accesses to the joystick, will be running at 4MHz if the user boots Raisin in accelerated mode. Access to Raisin's joystick port is always done at synchronous speed, 1.023 MHz. However, the neighboring machine cycles being executed four times faster than they used to be can cause time-dependent routines, such as joystick reading, to malfunction.

Asking the user to boot in normal mode is one option, but if the user insists on booting in accelerated mode (to make a game more challenging, for example), the fast speed would require very, very accurate pressing of the joystick button, or your program could miss it.

A better solution is to make sure the program is tuned to run correctly at either speed. To do this use either the IIc Plus ROM `WAIT` routine, or the revised version of it that follows. Both access a softswitch for a slot that has been set for synchronous, 1.023 MHz operation. However, if, as explained in the IIc Plus Pre-release Technical Note #4, your program has reconfigured slots 1 and 2 so that thier softswitches do not start the 50ms slow down timer, you should use the revised version of the IIc Plus ROM `WAIT` routine listed below. The routine is the same except that the softswitch accessed to start the 50ms slow down timer is $C0D0, which is guaranteed to always start a 50ms slow down timer on the IIc Plus. As with the original Apple II `WAIT` routine, the ROM `WAIT` routine and the `WAIT` routine below guarantee a minimum rather than absolute wait time.

```
                X6502                   ;enable 65C02 instructions
romwait         equ     $fca8
adamwait        jsr     sysid           ;<- jsr to here instead of to $fca8
                beq     newwait
                jmp     romwait

idloc1          equ     $FBB3
idloc2          equ     $FBC0
idloc3          equ     $FBBF
idval1          equ     $06
idval2          equ     $00
idval3          equ     $05
sysid           ldx     idloc1          ;check id bytes
                cpx     #idval1
                bne     idret
                ldx     idloc2
                bne     idret           ;skip cpx #0 since flag already set after ldx
                ldx     idloc3
                cpx     #idval3         ;found all the bytes? then z flag =1,else z=0
idret           rts                     ;20 cycles to id system if adam ant,$fbbf = 5

kbd             equ     $C000
newwait         phy                     ;wait routine
                ldy     #$D0            ;$C0D0 guaranteed 50ms slow down on
                phx                     ;adam ant
                sec
                tax
nw1             equ     *               ;first pass may be accelerated until
                lda     kbd,y           ;<-this instruction starts slow down
                txa                     ;min delay = 1/2(50+25A+5A^2)+29 cycles
nw2             equ     *               ;min delay = (A - 41) fewer cycles
                sbc     #$01            ;than standard wait
                bne     nw2             ;see Apple II Misc. Tech Note #12
                dex                     ;for further details on wait routine
                bne     nw1
                plx
                ply
                rts
```

Section 6: Identifying the //c Plus

The following routine shows the correct method for identifying the //c Plus.

```
SOURCE   FILE #01 =>IICPLUSID
0000:        0000    1           X6502
----- NEXT OBJECT FILE NAME IS IICPLUSID.0
3000:        3000    2           org   $3000
3000:        C082    3 ROMIN     equ   $C082
3000:        FBB3    4 idloc1    equ   $FBB3
3000:        FBC0    5 idloc2    equ   $FBC0
3000:        FBBF    6 idloc3    equ   $FBBF
3000:        0006    7 idval1    equ   $06
3000:        0000    8 idval2    equ   $00
3000:        0005    9 idval3    equ   $05
3000:AD 82 C0       10 sysid     lda   ROMIN          ;ROM must be enabled
3003:AD B3 FB       11           lda   idloc1         ;check id bytes
3006:C9 06          12           cmp   #idval1
3008:D0 10   301A   13           bne   negid
300A:AD C0 FB       14           lda   idloc2
300D:C9 00          15           cmp   #idval2
300F:D0 09   301A   16           bne   negid
3011:AD BF FB       17           lda   idloc3
3014:30 04   301A   18           bmi   negid          ;adjust for orig //c = $ff
3016:C9 05          19           cmp   #$05
3018:B0 01   301B   20           bcs   idret          ;carry set if //c+, clear otherwise
301A:18             21 negid     clc
301B:60             22 idret     rts                  ;carry set if //c+, clear otherwise
FBB3 IDLOC1         FBC0 IDLOC2         FBBF IDLOC3         301B IDRET
   06 IDVAL1           00 IDVAL2        ?  05 IDVAL3         301A NEGID
 C082 ROMIN          ?3000 SYSID
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 20-OCT-88 21:28
** TOTAL LINES ASSEMBLED    22
** FREE SPACE PAGE COUNT   84
```